# Programming Fundamentals with Python: Exam Preparation

Link to Judge: <https://judge.softuni.org/Contests/4778/Exam-Preparation-20-March-2024>

## 01. Registration

*John wants to register on a sales site, but too many people call him John. It asks you to write a program that will generate an original username.*

On the first line, you will **receive** **the username** that he wants to use in the first place. On the following lines, you will receive **commands** until the **"Registration"** command. There are **five** possible commands:

* **"Letters {Lower/Upper}"**
  + **Replace** all letters with lower case or with upper case, then **print** the result.
* "**Reverse {startIndex} {endIndex}"**
  + **Reverse** the substring from the start indexuntil the end index (both inclusive), then **print** it. Do **NOT** change it in the username.

**Note:** Check if the indices are valid. If they aren't - skip the command. An index is valid when it is **non-negative** and **less than the size** of the collection.

* "**Substring {substring}"**
  + If the username **contains** **the** **given substring**, **cut** it out and **print** the result without the cut substring.
  + Otherwise, print:

**"The username {string} doesn't contain {substring}."**

* "**Replace {char}"**
  + **Replace** all occurrences of the **given char** with a **dash** (**-**) and **print** the result.
* "**IsValid {char}"**
  + For a username to be **valid,** it must contain **the given char**.If it is, print **"Valid username."**.
  + Otherwise, print: **"{char} must be contained in your username."**

### Input

* On the **first line,** you are going to receive the **string**.
* On the following **lines**, until the **"Registration"** command is received, you will receive commands.
* All commands are **case-sensitive**.

### Output

* **Print** the **output** of every **command** in the **format** **described** **above**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| John  Letters Lower  Substring SA  IsValid @  Registration | john  The username john doesn't contain SA.  @ must be contained in your username. |
| ThisIsSoftUni  Reverse 1 3  Replace S  Substring hi  Registration | sih  ThisIs-oftUni  TsIs-oftUni |

## 02. Destination Mapper

*Now that you have planned your tour, you are ready! Your next task is to mark all the points on the map that you are going to visit.*

You will be given a **string** representing some **places** on the map. You have to **filter** only the **valid ones**. A valid location is:

* Surrounded by **"="** or **"/"** on **both sides** (the **first** and the **last** symbols must **match**)
* After the **first "="** or **"/"** there should be **only letters** (the **first** must be **upper-case,** other letters could be upper or lower-case)
* The **letters** must be **at least 3**

**Example**: In the string **"=Hawai=/Cyprus/=Invalid/invalid==i5valid=/I5valid/=i="** only the **first two** locations are valid.

After you have **matched** all the **valid locations**, you have to **calculate travel points**. They are calculated by **summing** the **lengths** of all the **valid destinations** that you have found on the map.

In the end, on the **first line,** print: **"Destinations: {destinations joined by ', '}"**.

On the **second line,** print **"Travel Points: {travel\_points}"**.

### Input / Constraints

* You will receive a string representing the locations on the map.
* JavaScript: you will receive a single parameter: string.

### Output

* Print the **messages described above.**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| =Hawai=/Cyprus/=Invalid/invalid==i5valid=/I5valid/=i= | Destinations: Hawai, Cyprus  Travel Points: 11 |
| ThisIs some InvalidInput | Destinations:  Travel Points: 0 |

## 03. Bakery Shop

*Maria is opening a bakery shop, and she needs your help to keep track of the stock availability.*

Create a program that keeps the information about the stock at the shop.

You will be receiving **lines** with commands until you receive the **"Complete"** command. The **possible** commands are:

* **"Receive {quantity} {food}":**
  + Add the **quantity** to the given **food**.
  + If the food **does not** exist, **add** it to your record.
  + If the **quantity is invalid** (<= 0), ignore the command.
* **"Sell {quantity} {food}":**
  + If the **food** is **not in your record**, print: **"You do not have any {food}."**.
  + If there is **not enough quantity** of the wanted food, you should sell (decrease) **what you** **have in stock** and then remove **the food** from your record. Print: **"There aren't enough {food}. You sold the last {sold quantity} of them."**
  + Otherwise, **decrease** the **quantity** of the given **food** and print: **"You sold {quantity} {food}."**.If, **after reducing the quantity**, there is **0 amount of this food**, you should **remove it** from your record.

**You must keep track of all sold food quantities!**

In the end, you should **print the stock availability** in the format:

**"{food1}: {quantity}**

**{food2}: {quantity}**

**…**

**{foodN}: {quantity}**

**All sold: {count of all sold food quantity} goods"**

### Input

* You will be receiving linesuntil you receive the **"Complete"** command.
* The input will **always** be **valid**.

### Output

* **Print** the **stock availability** in the **format** described above.
* **Print** the amount of **all sold food** in the **format** described above.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Receive 105 cookies  Receive 10 donuts  Sell 10 donuts  Sell 1 bread  Complete | You sold 10 donuts.  You do not have any bread.  cookies: 105  All sold: 10 goods |
| Receive 10 muffins  Receive 23 bagels  Sell 5 muffins  Sell 10 bagels  Complete | You sold 5 muffins.  You sold 10 bagels.  muffins: 5  bagels: 13  All sold: 15 goods |